

# Recognizing Phrases with Filtering-Ranking Perceptron

Xavier Carreras and Lluís Màrquez



Centre de Tecnologies i Aplicacions del Llenguatge i la Parla

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# Outline

- Phrase Recognition
- Filtering-Ranking Strategy
- FR-Perceptron
- Experiments

# Phrase Recognition Problems

- Following CoNLL Shared Tasks:
  - ★ 2000: Chunking of Syntactic Base Phrases
  - ★ 2001: Identification of Syntactic Clauses
  - ★ 2002, 2003: Named Entity Extraction
  - ★ 2004: Recognition of Semantic Roles
- Also, Full Syntactic Parsing

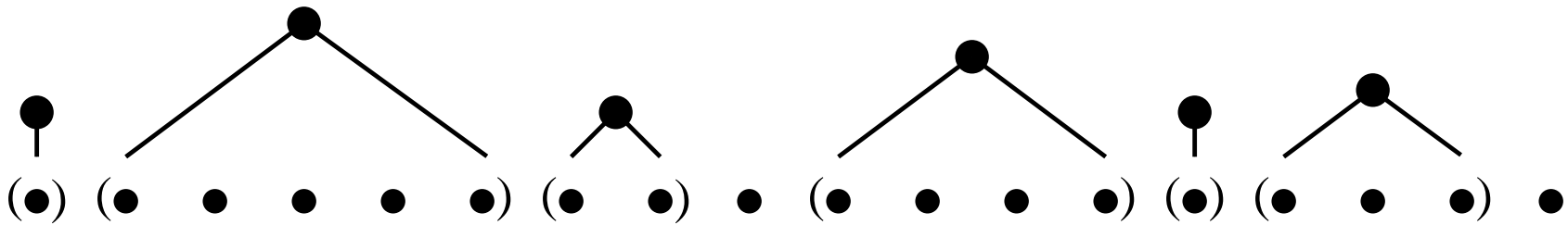
# Phrase Recognition in CoNLL: Example

WORDS	PoS	CHUNKS	CLAUSES	NE	<-- SEMANTIC ROLES ----->	
The	DT	B-NP	(S*	0	(AO*	*
San	NNP	I-NP	*	B-ORG	*	*
Francisco	NNP	I-NP	*	I-ORG	*	*
Examiner	NNP	I-NP	*	I-ORG	*AO)	*
issued	VBD	B-VP	*	0	(V:ISSUE *V)	*
a	DT	B-NP	*	0	(A1*	(A1*
special	JJ	I-NP	*	0	*	*
edition	NN	I-NP	*	0	*A1)	*A1)
around	IN	B-PP	*	0	(AM-TMP*	*
noon	NN	B-NP	*	0	*AM-TMP)	*
yesterday	NN	B-NP	*	0	(AM-TMP*AM-TMP)	*
that	WDT	B-NP	(S*	0	(C-A1*	(R-A1*R-A1)
was	VBD	B-VP	(S*	0	*	*
filled	VBN	I-VP	*	0	*	(V:FILL *V)
entirely	RB	B-ADVP	*	0	*	(AM-MNR*AM-MNR)
with	IN	B-PP	*	0	*	*
earthquake	NN	B-NP	*	0	*	(A2*
news	NN	I-NP	*	0	*	*
and	CC	I-NP	*	0	*	*
information	NN	I-NP	*S)S)	0	*C-A1)	*A2)
.	.	0	*S)	0	*	*

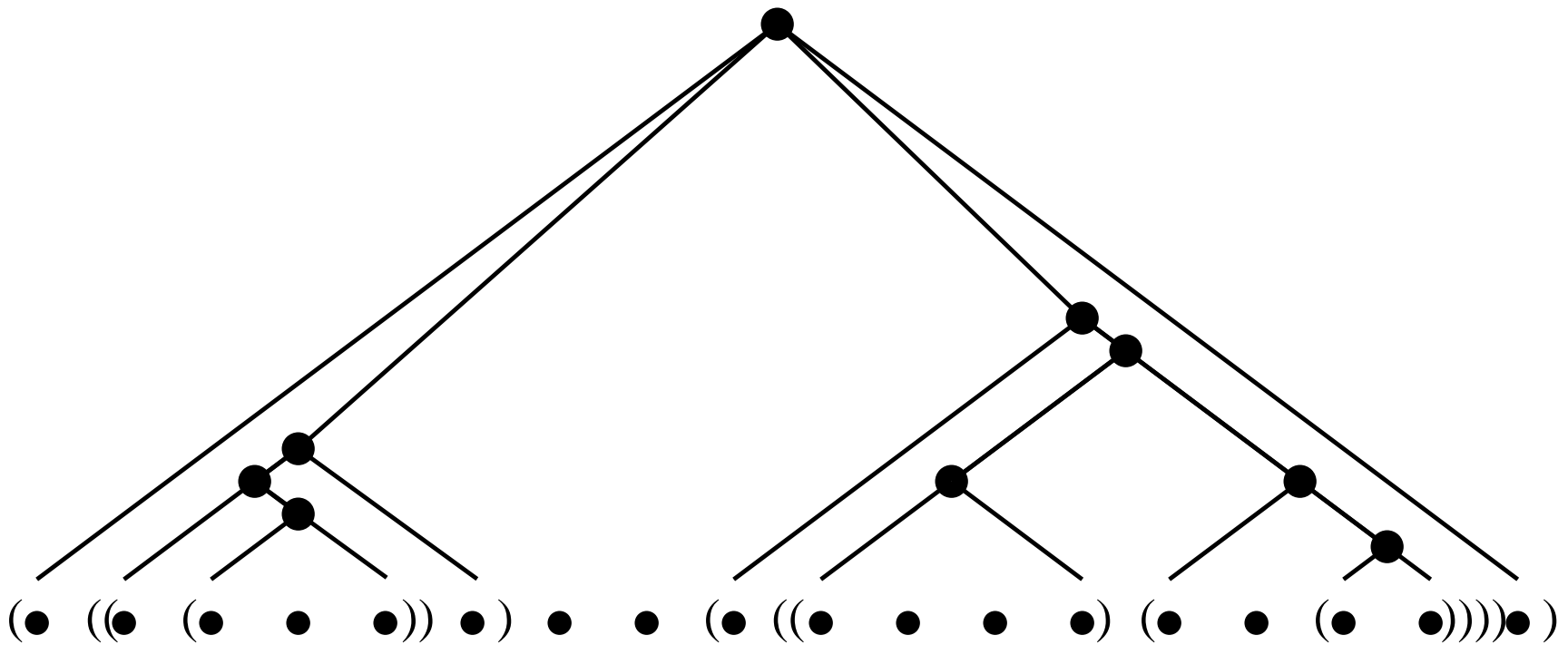
# Phrase Recognition: general

- Goal: find phrases in a sentence, of types in  $\mathcal{K}$ .
- Solution: a set of phrases, each of the form  $(s, e)_k$ , satisfying some constraints:
  - ★ Phrases do not overlap (do not cross boundaries).
  - ★ Sequential Structures: phrases do not embed.
  - ★ Hierarchical Structures: phrases may be embedded.
- Evaluation: Precision/Recall/ $F_1$  of recognized phrases.

# Sequential Phrase Recognition: schematic view

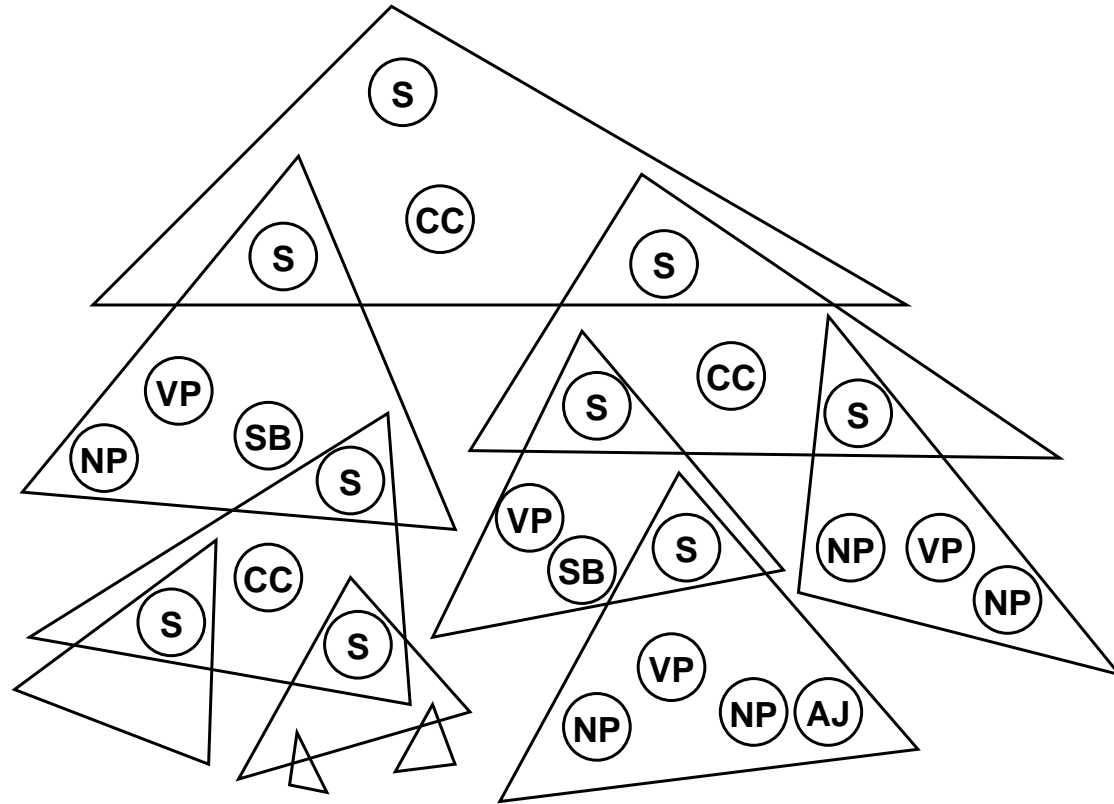


# Hierarchical Phrase Recognition: schematic view





## Observation 2: Recursive Structures



Desirable to put learning in high-order level.

# Recognizing Structures: General Approach

- Decompose global problems into *tractable* and *learnable* subproblems:
  - ★ Chunking: as tagging, with BIO, Open-Close, . . .
  - ★ Hierarchies: CKY-style decisions, shift-reduce, . . .
- Design decoder algorithm which *infers* global solution given the local predictions: greedy, beam search, dynamic programming, . . .
- Learn local functions for each local subproblem.

# Recognizing Structures: Learning

- Local learning: each local function is trained separately, as a (binary) classification algorithm.
  - ★ Good understanding on learning classifiers.
  - ★ *but* local accuracies don't guarantee global accuracy (after inference).
  - ★ *that is*, a local classification behavior might not be the optimal within the decoder.
- Global learning: train the Recognizer as a composed function for the final problem.

# Outline

- Phrase Recognition
- Filtering-Ranking Strategy
- FR-Perceptron
- Experiments

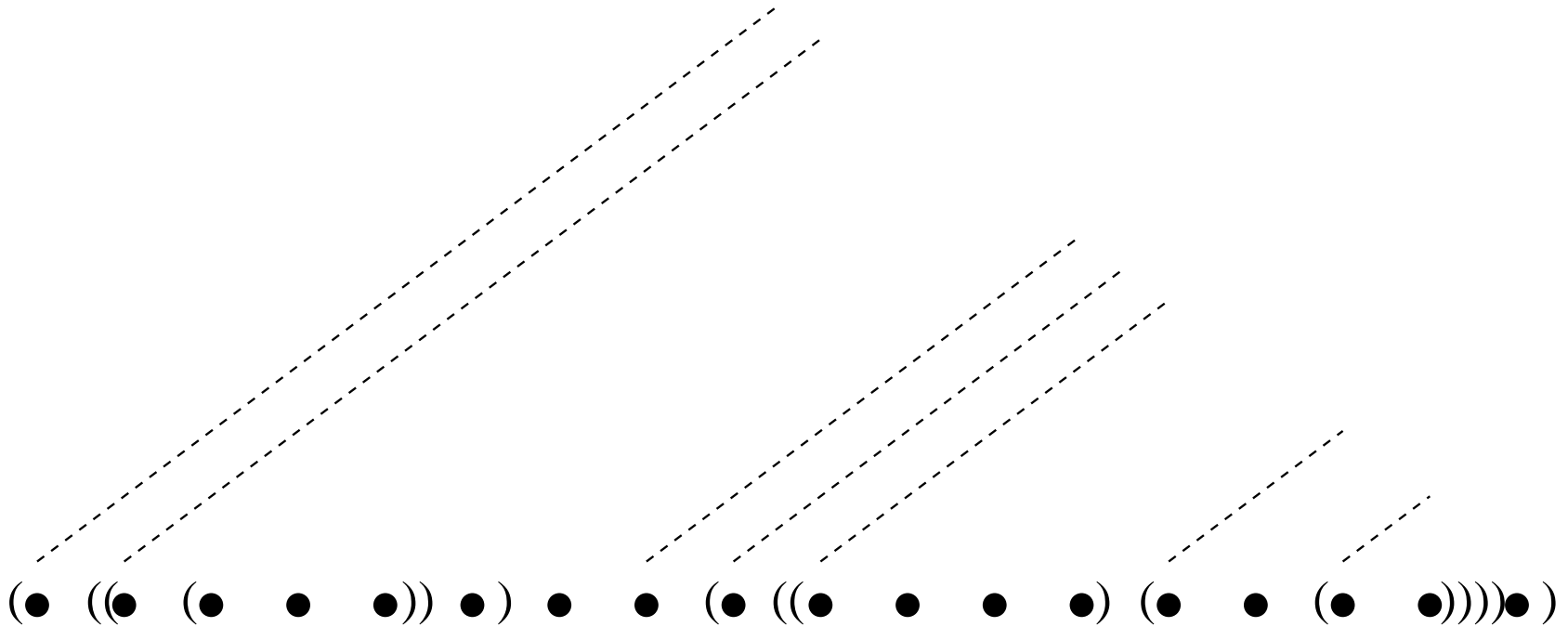
# Filtering-Ranking Strategy (i)

1. Generate phrase candidates with Start-End functions.
2. Score each phrase candidate.
3. Build the hierarchy with best phrases.

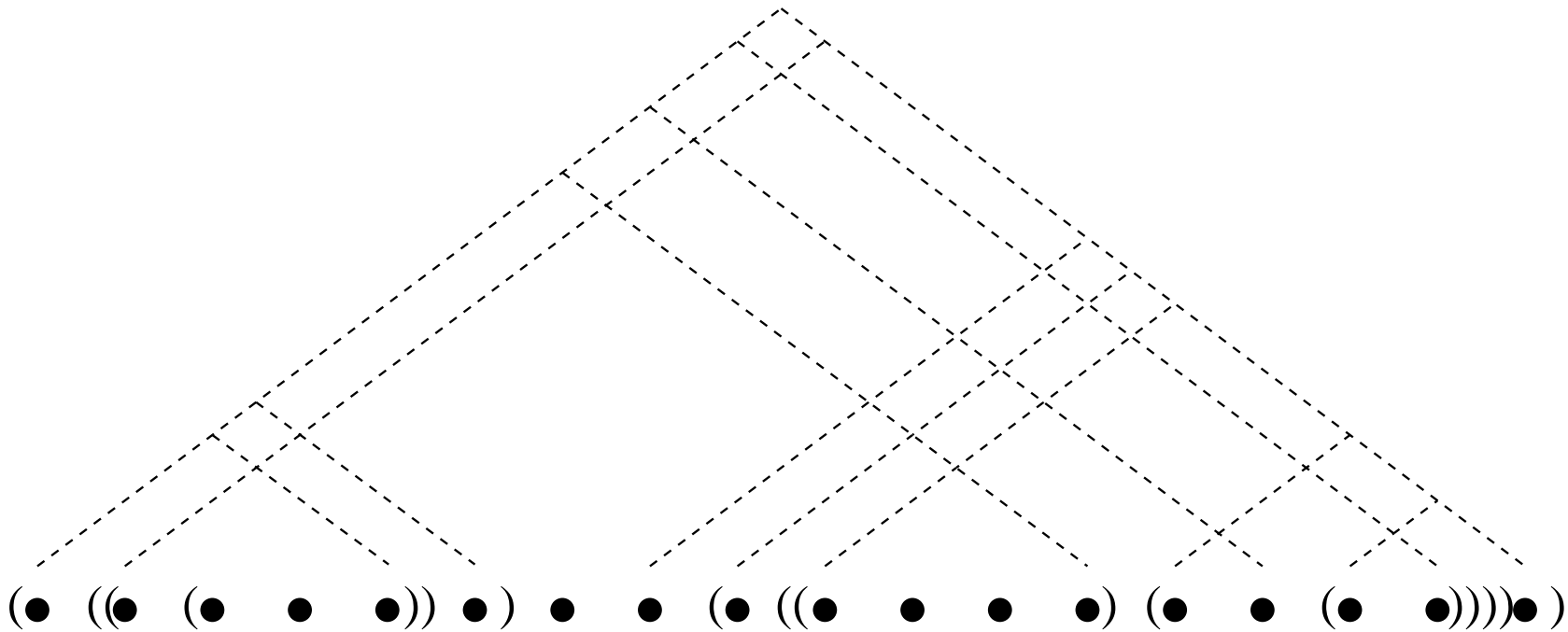
# Filtering-Ranking Strategy (ii)

(• ((• (• • •)) •) • • (• ((• • • •) (• • (• •))))•)

# Filtering-Ranking Strategy (ii)

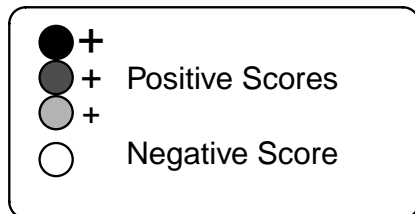
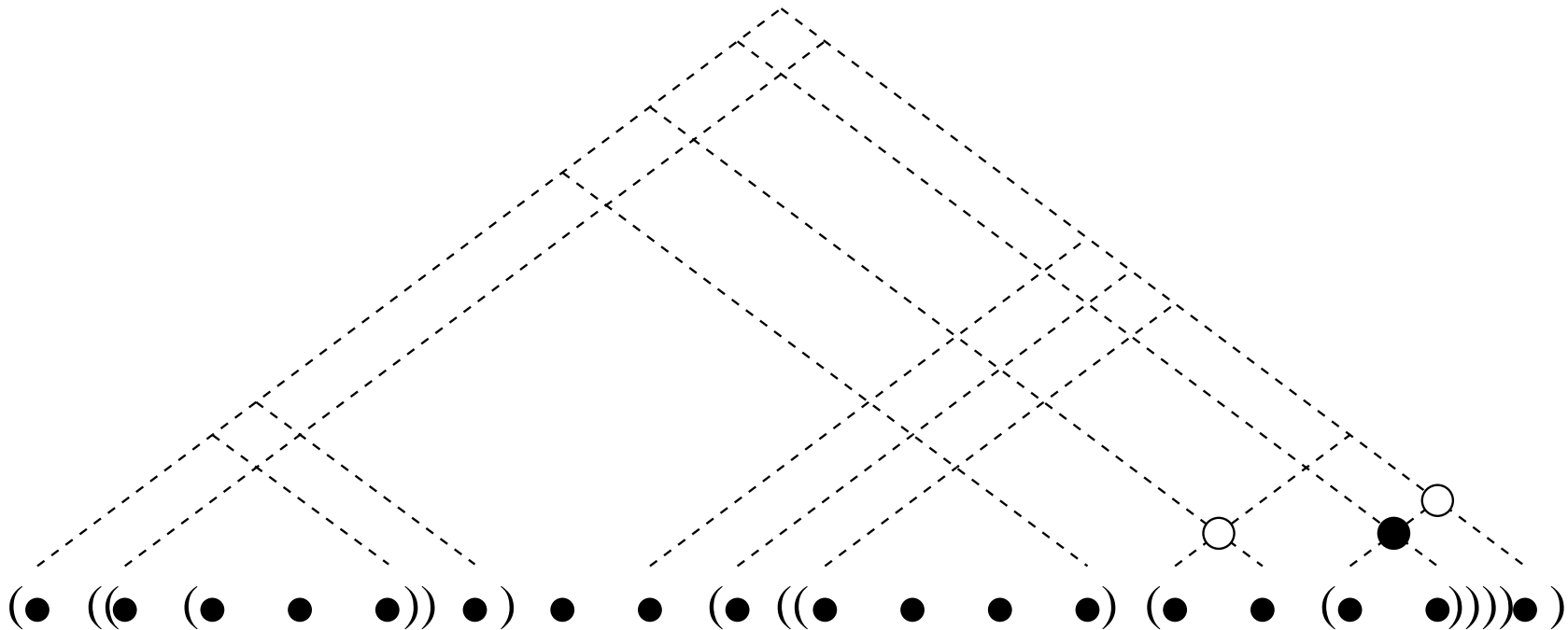


# Filtering-Ranking Strategy (ii)



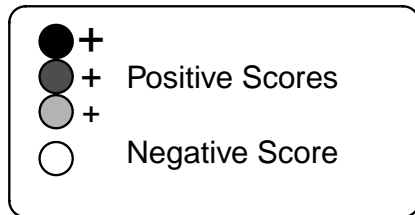
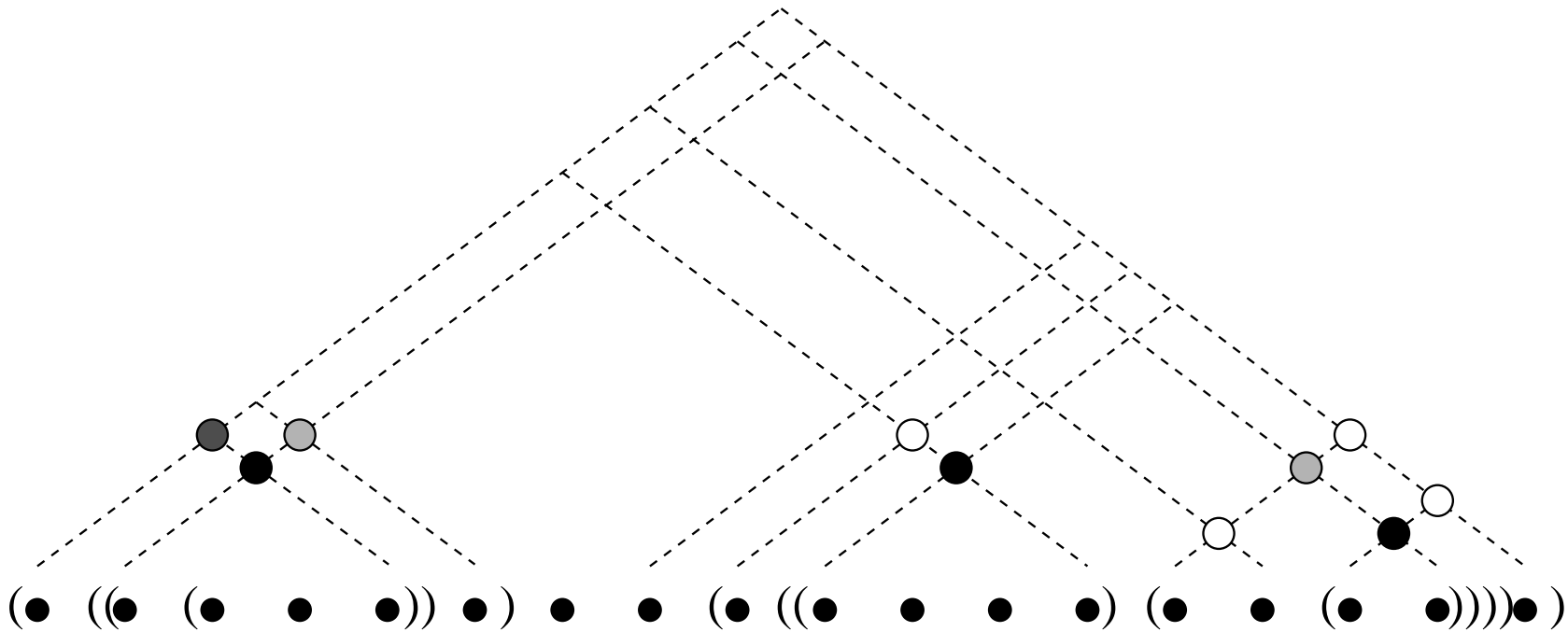


# Filtering-Ranking Strategy (ii)

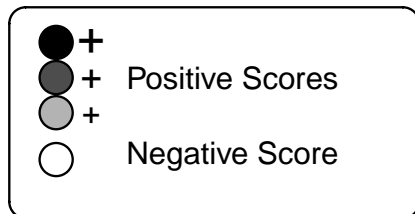
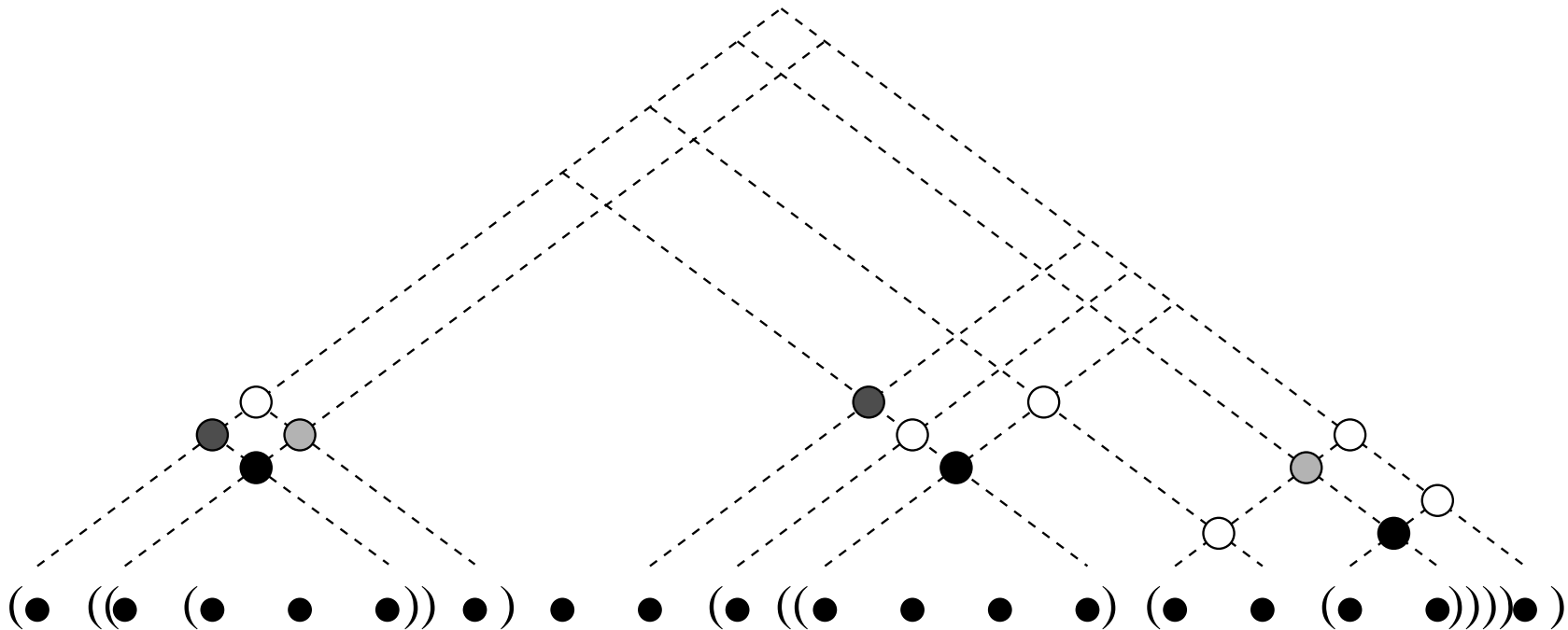




# Filtering-Ranking Strategy (ii)



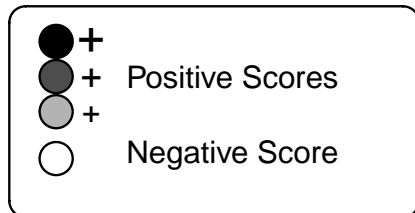
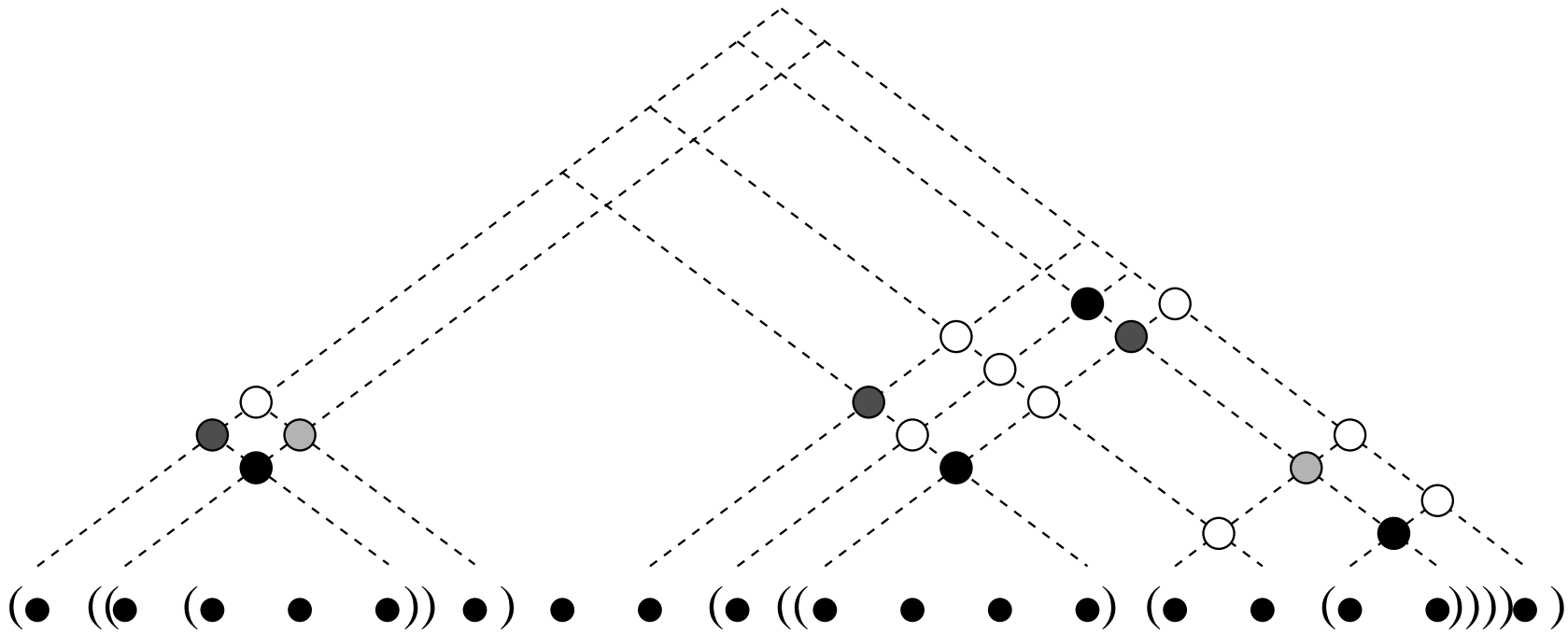
# Filtering-Ranking Strategy (ii)





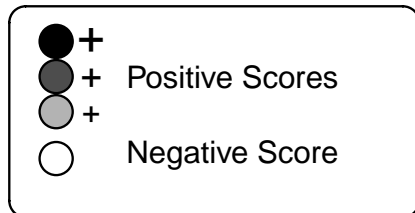
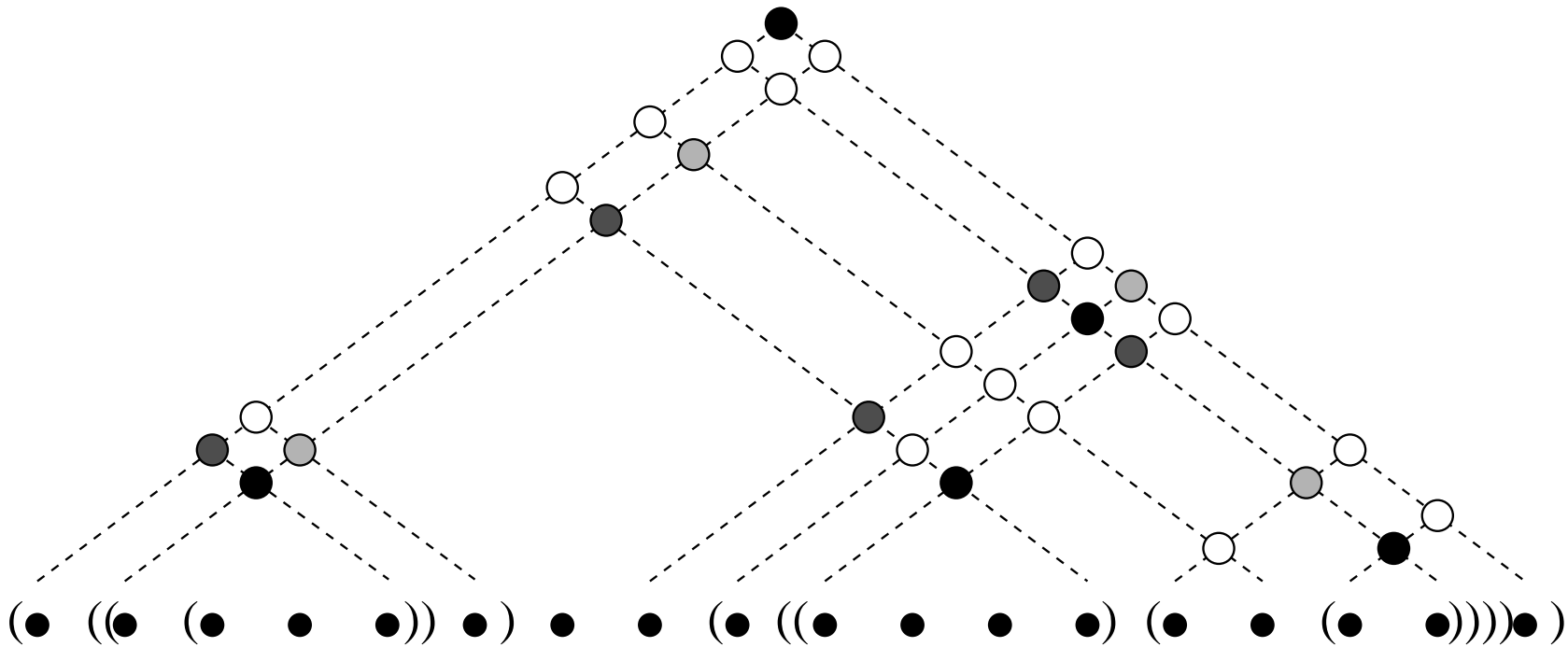


# Filtering-Ranking Strategy (ii)



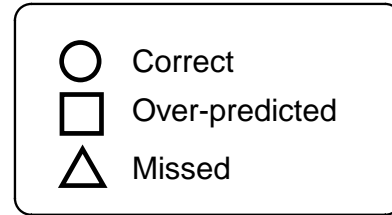
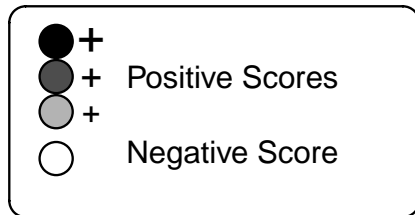
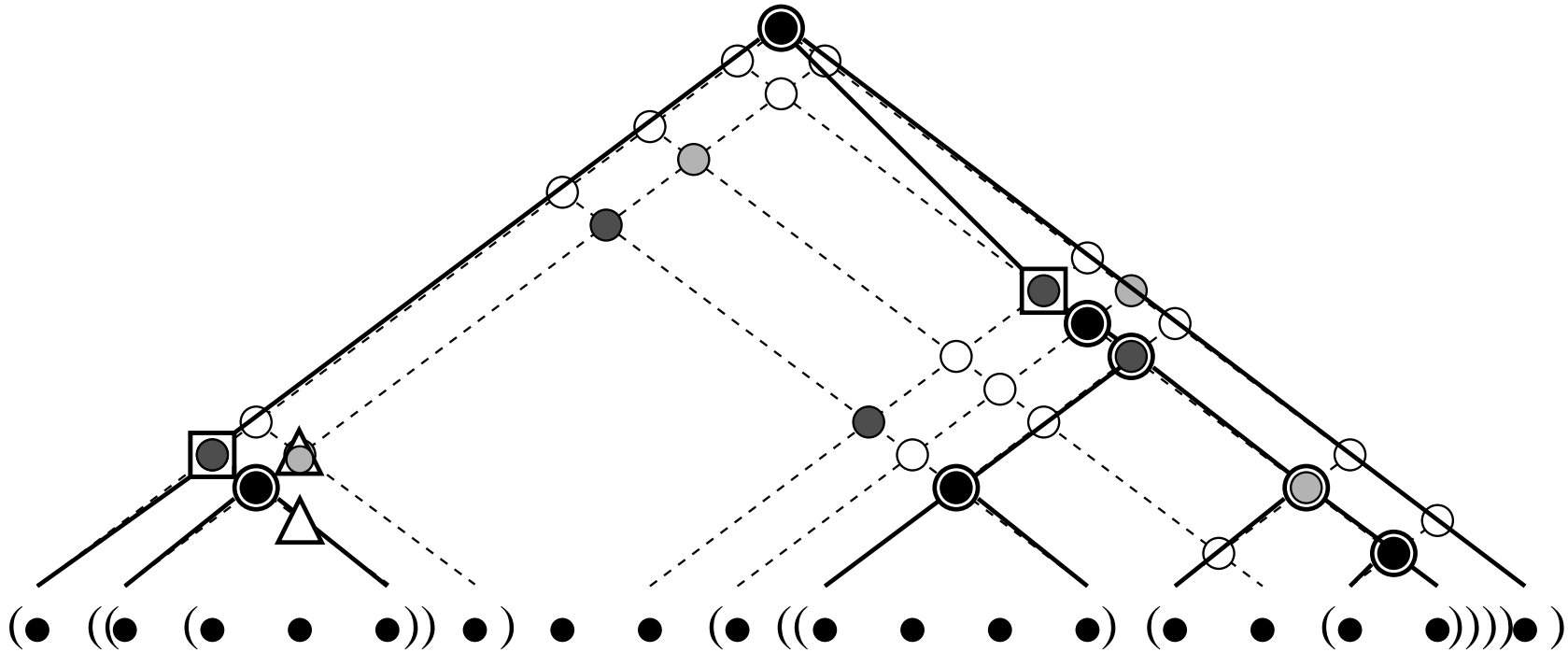


# Filtering-Ranking Strategy (ii)





# Filtering-Ranking Strategy (ii)



# Filtering-Ranking Strategy (iii)

$\mathcal{Y}$ : **solution space**, i.e. set of all coherent phrase sets.

$\mathcal{Y}_{SE}$ : **practical solution space**, filtered at word level.

$$R(x) = \arg \max_{y \in \mathcal{Y}_{SE}} \sum_{(s,e)_k \in y} \text{score}((s,e)_k, x, y_{s:e})$$

$$\mathcal{Y}_{SE} = \{y \in \mathcal{Y} \mid \forall (s,e)_k \in y \text{ start}^k(s) \wedge \text{end}^k(e)\}$$

- Sequential case:  $O(n^2)$  Dynamic Prog. search
- Hierarchical case:  $O(n^3)$  Dynamic Prog. search

# Outline

- Phrase Recognition
- Filtering-Ranking Strategy
- FR-Perceptron
- Experiments

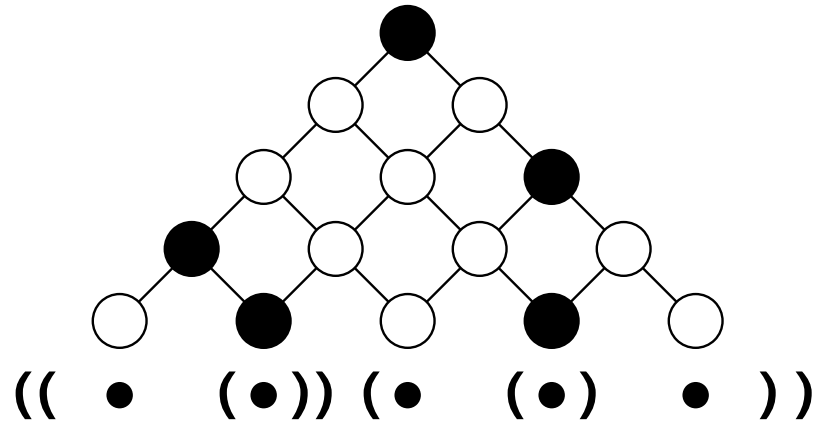
# Learning Challenges

- Learn all functions  $(\text{start}^k, \text{end}^k, \text{score}^k)$  so as to maximize the  $F_1$  measure on the recognition of phrases.
- Start-End:
  - ★ As filters, rather than classifiers.
  - ★ They define the input space to the score functions
- Score functions:
  - ★ The negative space is too big  $\sim O(n^2)$ .
  - ★ We need to know about Start-End behavior.
  - ★ As Rankers, rather than Classifiers.

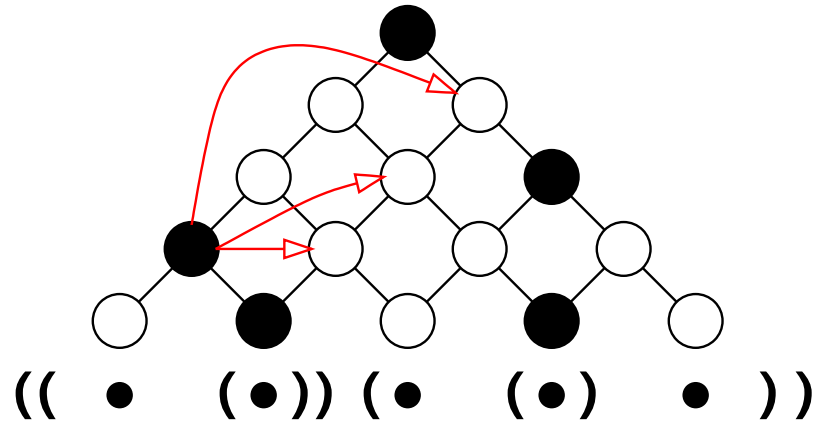




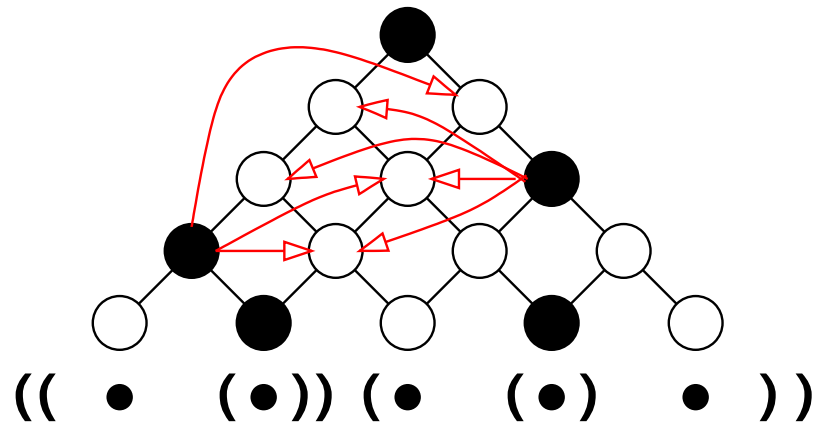
# Score as Ranker: Pairwise Order Relations



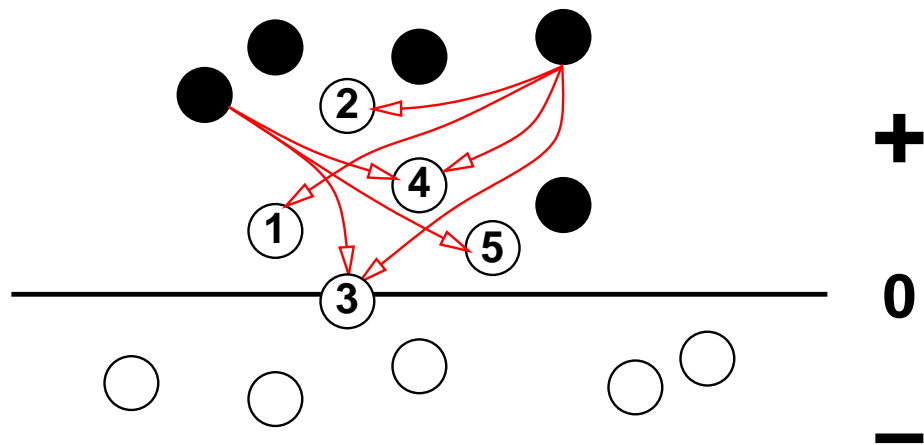
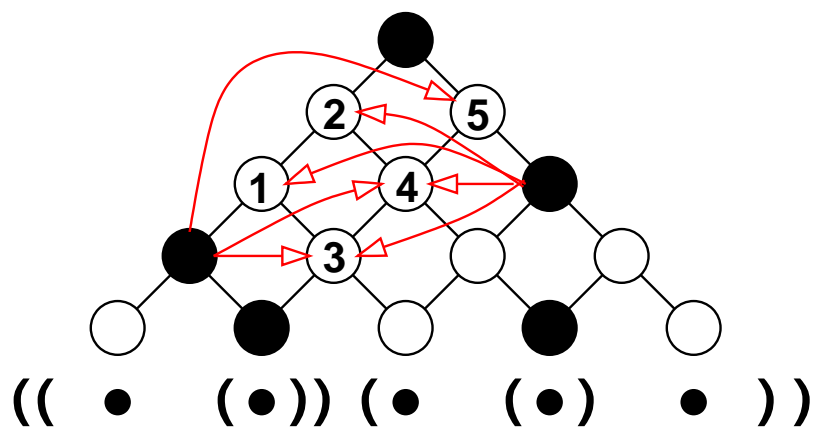
# Score as Ranker: Pairwise Order Relations



# Score as Ranker: Pairwise Order Relations

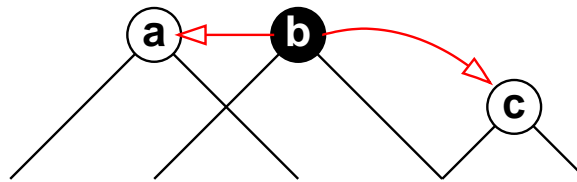


# Score as Ranker: Pairwise Order Relations



# Score as Ranker: Pairwise Order Relations Counterexample

- Pairwise Order Relations are not enough:



It may happen that :

- ★  $\text{score}(b) > \text{score}(a)$  and  $\text{score}(b) > \text{score}(c)$
- ★ but  $\text{score}(a) + \text{score}(c) > \text{score}(b)$

# Score as Ranker: Sentence Level Learning

- We follow Collins' approach (EMNLP 2002):
  - Guide learning at the sentence-level
- Two key points:
  - ★ Mistake-driven learning, a.k.a. Perceptron
  - ★ Learn from the output of the inference
- Our contribution (Carreras and Màrquez, NIPS 2003):
  - We propagate mistakes to the filtering layer

# Filtering-Ranking Perceptron

- All functions are learned together, while visiting online training sentences.
- Algorithm: Given a sentence:
  1. Predict the phrase hierarchy.
  2. Identify errors and provide feedback.
    - We consider only errors at global level:
      - ★ Missed Phrases
      - ★ Over-predicted Phrases

## Feedback on Missed prases

If a phrase  $(s, e)_k$  is missed, do **promotion** updates:

- Boundaries:

$$\text{if } (\mathbf{w}_S \cdot \phi_w(x_s) \leq 0) \text{ then } \mathbf{w}_S = \mathbf{w}_S + \phi_w(x_s)$$

$$\text{if } (\mathbf{w}_E \cdot \phi_w(x_e) \leq 0) \text{ then } \mathbf{w}_E = \mathbf{w}_E + \phi_w(x_e)$$

- Score :

$$\text{if } (\mathbf{w}_S \cdot \phi_w(x_s) > 0 \wedge \mathbf{w}_E \cdot \phi_w(x_e) > 0) \text{ then}$$

$$\mathbf{w}_k = \mathbf{w}_k + \phi_p(s, e)$$

# Feedback on Over-Predicted prases

If a phrase  $(s, e)_k$  is over-predicted, do **demotion** updates:

- Score :

$$\mathbf{w}_k = \mathbf{w}_k - \phi_p(s, e)$$

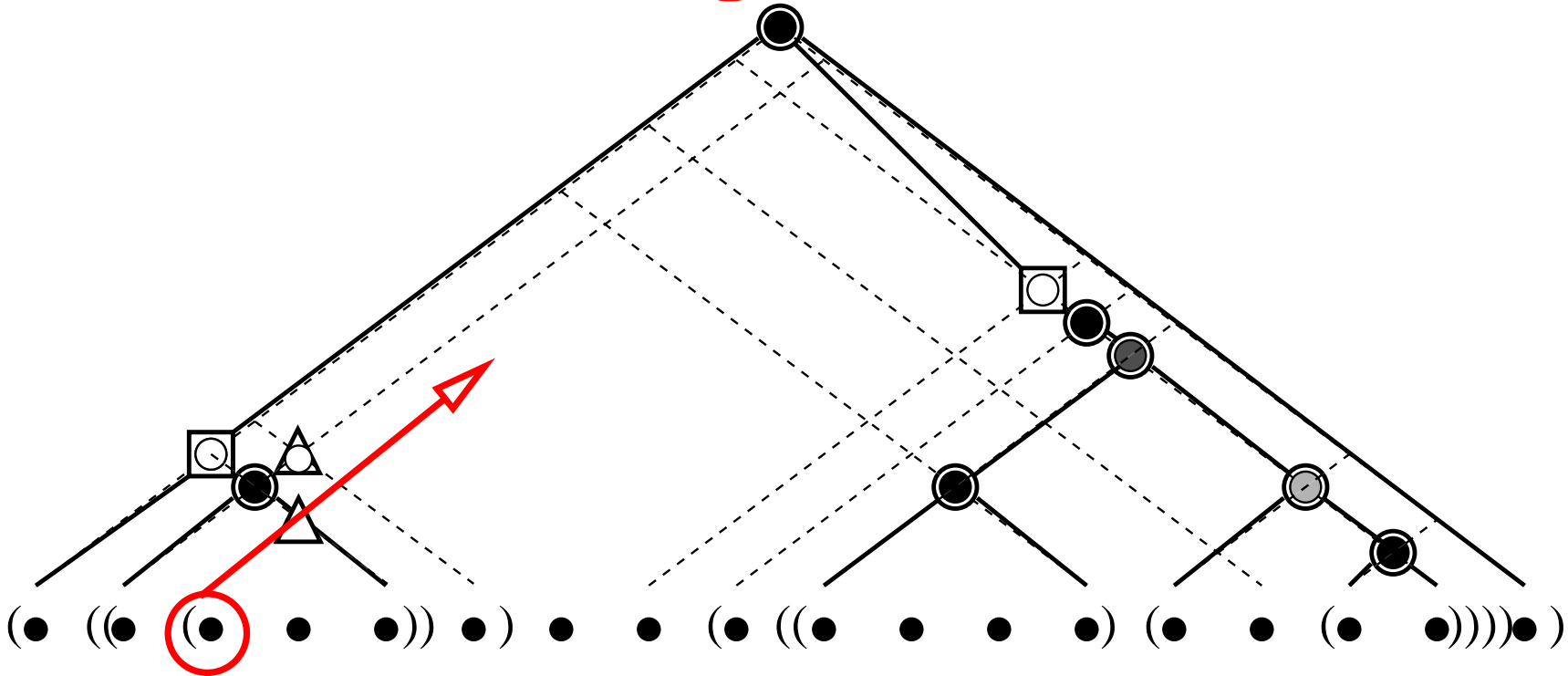
- Boundaries:

if  $(x_s \notin \text{STARTS}(y^*))$  then  $\mathbf{w}_S = \mathbf{w}_S - \phi_w(x_s)$

if  $(x_e \notin \text{ENDS}(y^*))$  then  $\mathbf{w}_E = \mathbf{w}_E - \phi_w(x_e)$



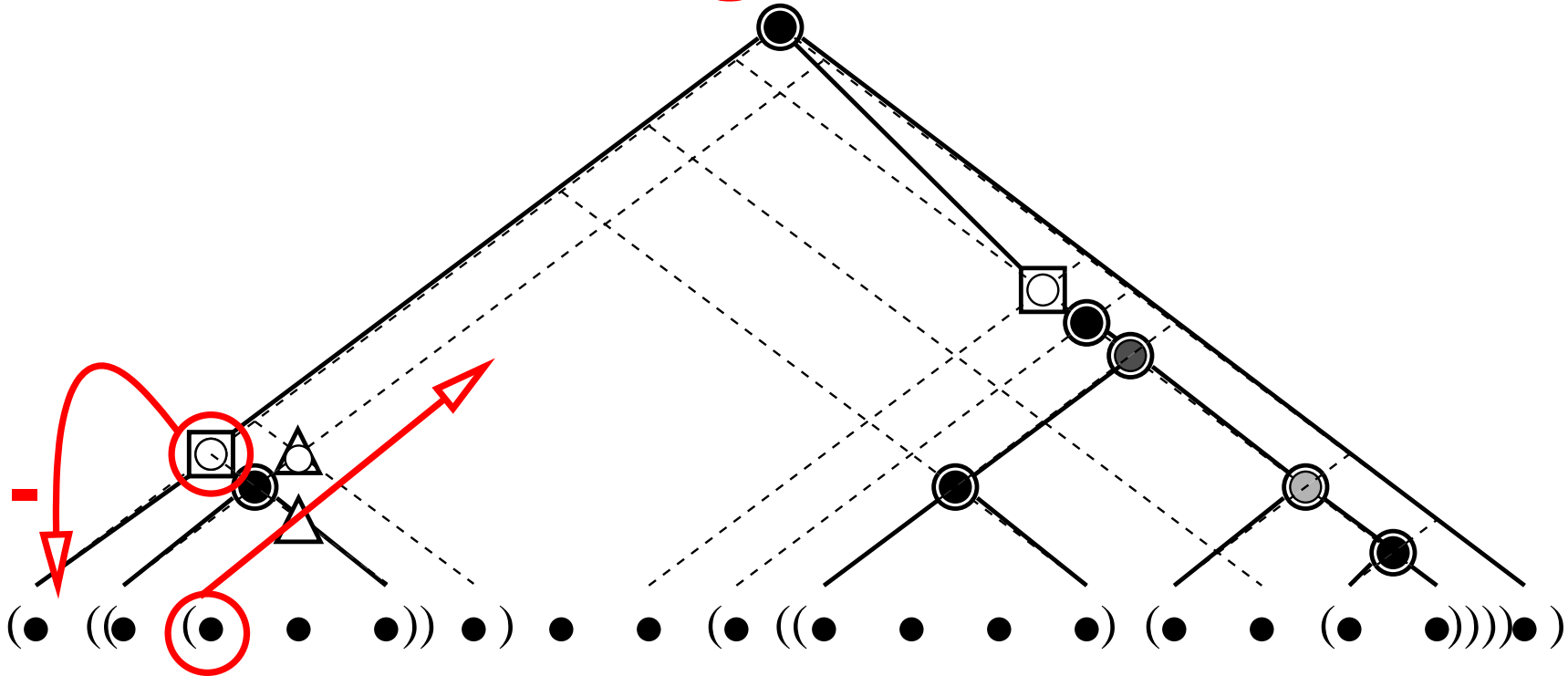
# Learning Feedback



●+	
●+	Positive Scores
●+	
○	Negative Score

○	Correct
□	Over-predicted
△	Missed

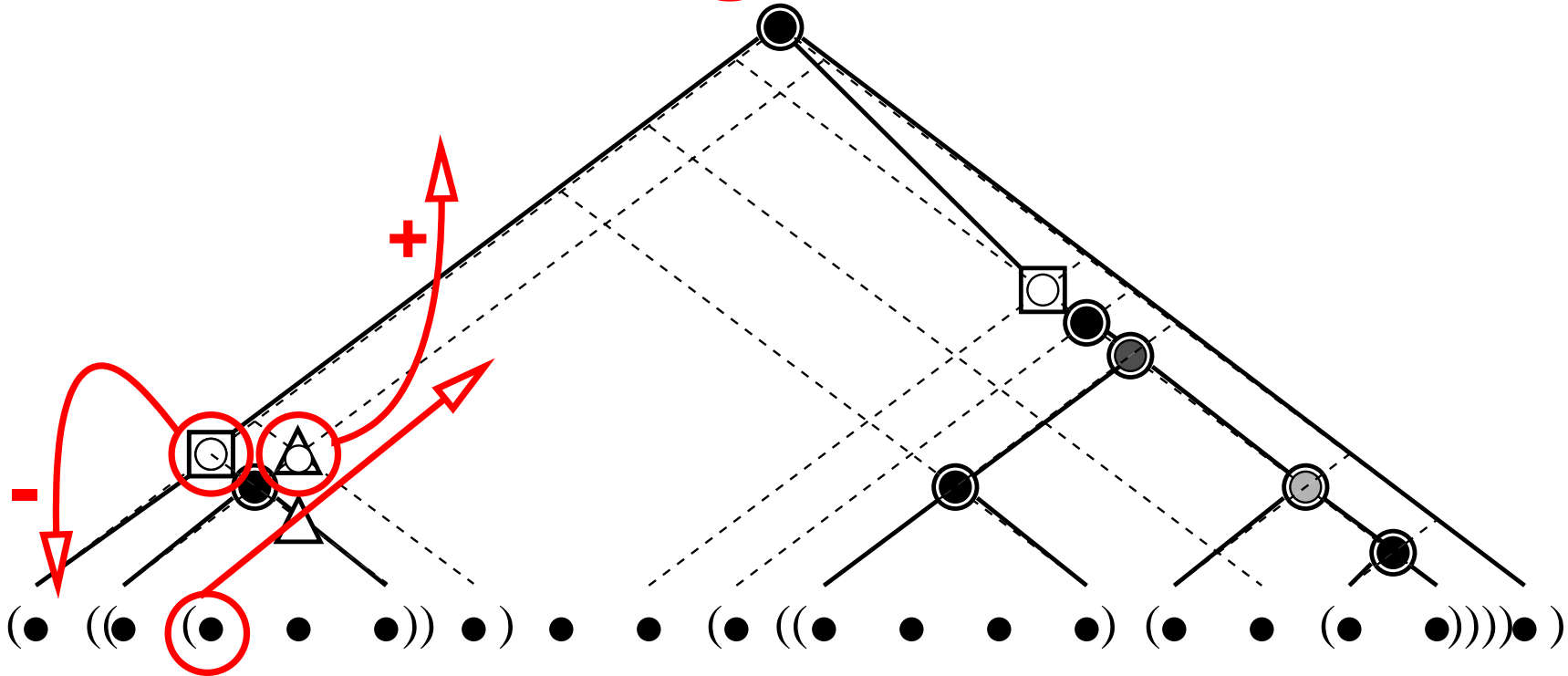
# Learning Feedback



●+	
●+	Positive Scores
●+	
○	Negative Score

○	Correct
□	Over-predicted
△	Missed

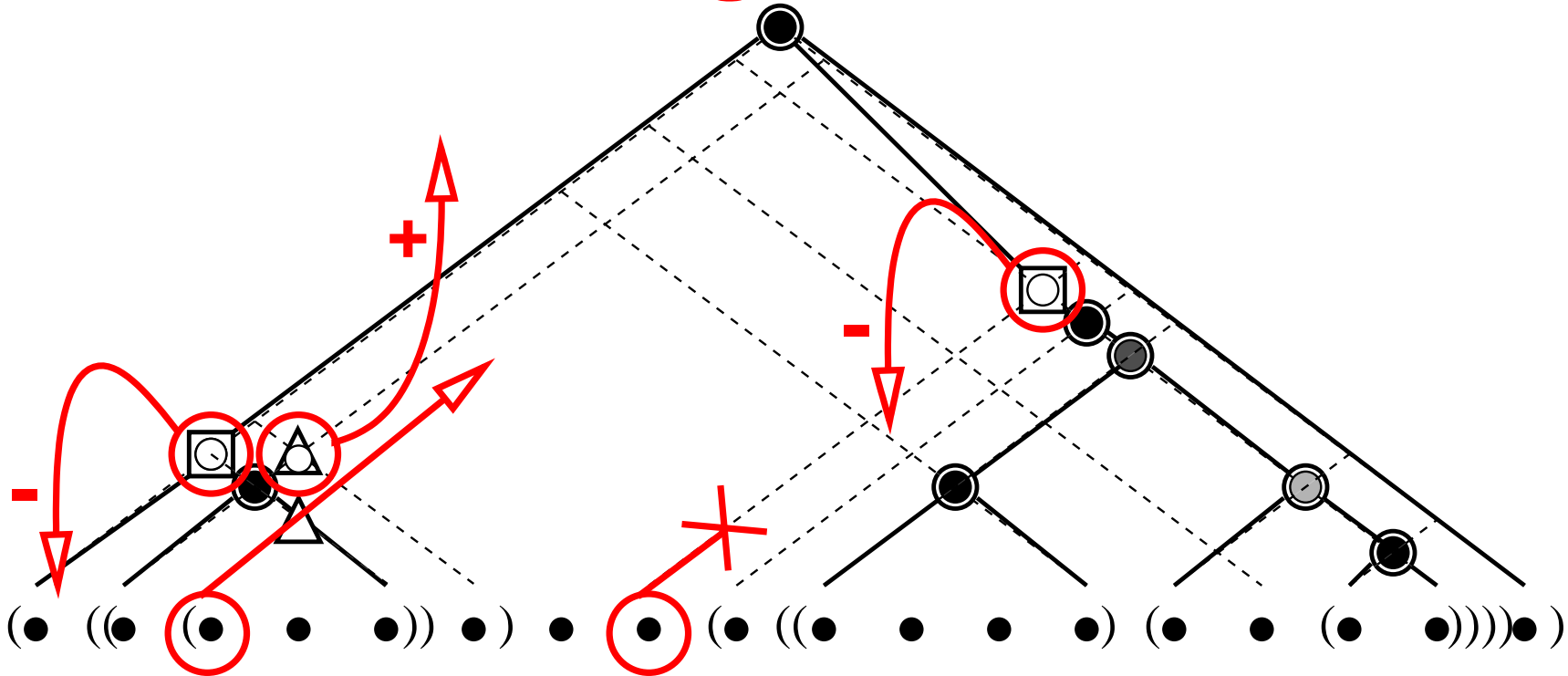
# Learning Feedback



● +  
● + Positive Scores  
● +  
○ Negative Score

○ Correct  
□ Over-predicted  
△ Missed

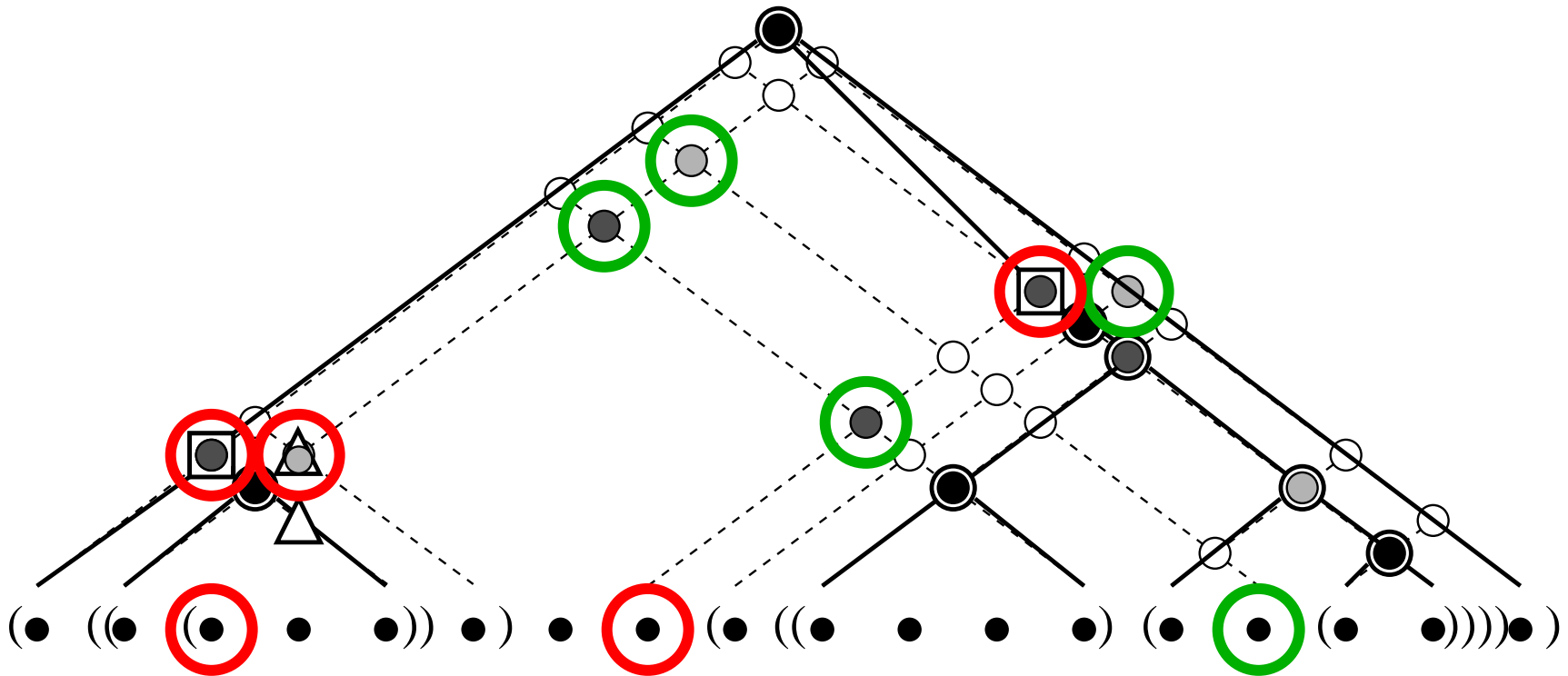
# Learning Feedback



● +
● + Positive Scores
● +
○ Negative Score

○ Correct
□ Over-predicted
△ Missed

# Conservative Feedback



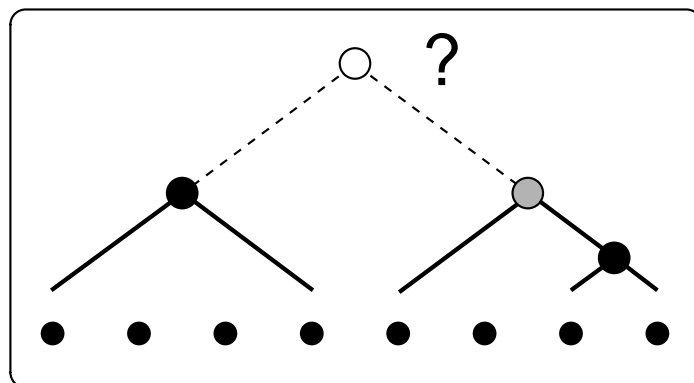
Local errors which do not hurt global performance are not penalized. (Crammer & Singer, 2003), (Har-Peled et al., 2002)

# On Representation

- Features on words: standard window-based features.
- Features on phrases:
  - ★ Windows at start/end boundaries.
  - ★ Patterns of the internal structure.

## On Representation: internal structure

When visiting a phrase, the internal structure is already computed:



Exploitation through patterns and constraints, via:

- Linguistically-motivated, grammar-based.
- Kernels, ie. exhaustive exploration of the structure.

# Final Architecture

- Exploration, incrementality, local decisions, inference.
- Learning Functions at word and phrase level.
- Learning Constraints at sentence level:
  - ★ All functions learned together, capturing interactions.
  - ★ Functions are modeled so as to optimize its behavior within the parser, i.e. as filters and rankers.
- Practical important tricks:
  - ★ Feature expansion with polynomial kernels,  $d = 2$ .
  - ★ Perceptron with averaged predictions.

# Outline

- Phrase Recognition
- Filtering-Ranking Strategy
- FR-Perceptron
- Experiments

# Experiments on Clause Identification

Given the same model (F&R) and the same learner (VP), we compare three training strategies:

**CB-VP** Classification feedback (0/1 Loss)

Each function trained separately, batch.

**CO-VP** Classification feedback (0/1 Loss)

Functions learned together, online.

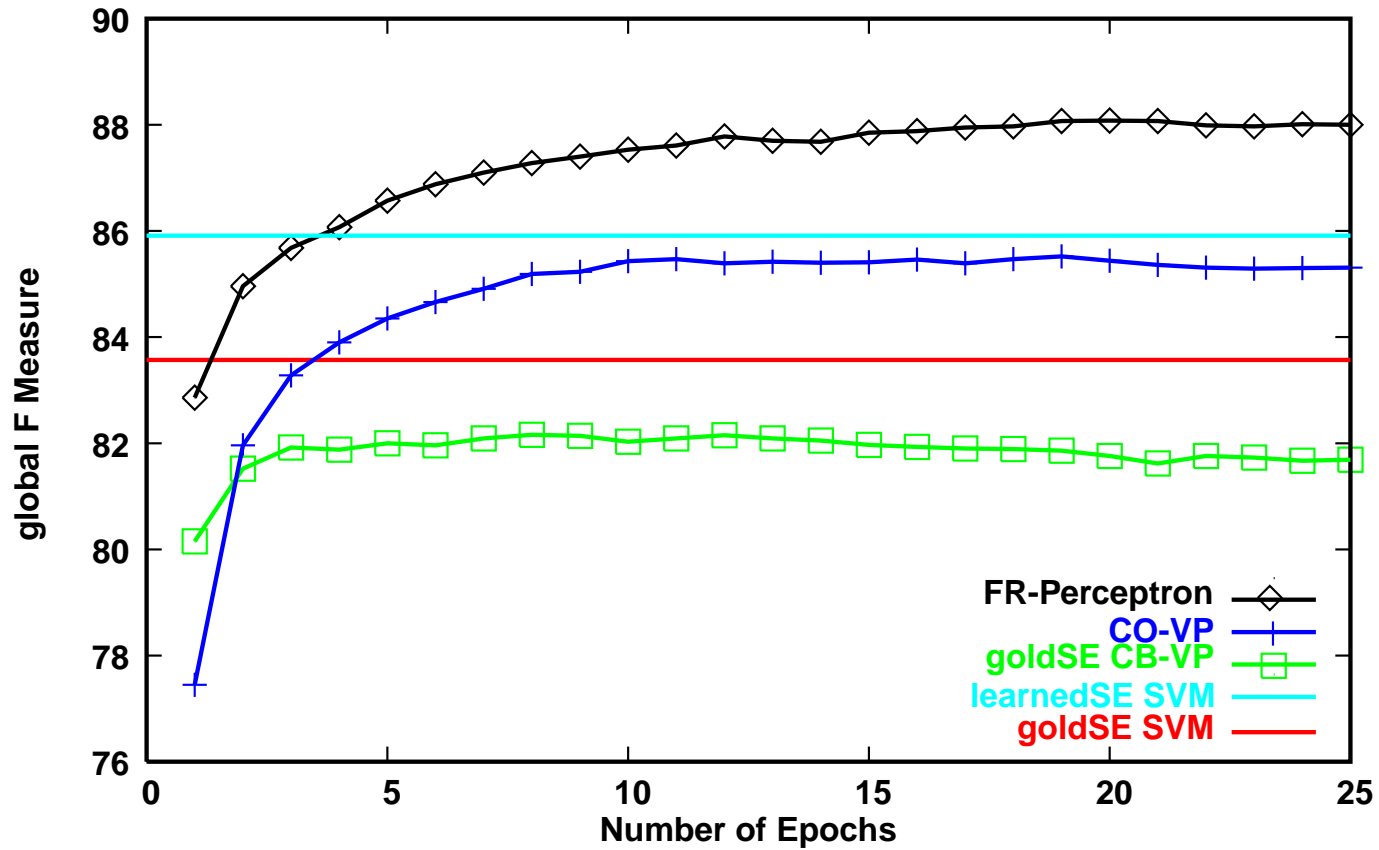
**F&R-VP** Conservative feedback (wrt.  $\arg \max$ )

Functions learned together, online.

## Training functions in batch setting

Algorithm	Generation	#Neg.	Precision	Recall	F <sub>1</sub>
CBVP	goldSE	26,374	83.84	80.55	82.16
SVM	goldSE	26,374	84.31	82.83	83.57
SVM	$\theta = 0$	28,165	88.14	82.85	85.41
SVM	$\theta = -0.3$	28,747	88.34	82.76	85.46
SVM	$\theta = -0.5$	29,145	88.46	82.61	85.43
SVM	$\theta = -0.7$	29,610	88.54	82.48	85.41
SVM	$\theta = -0.9$	30,432	88.91	82.58	85.63
SVM	$\theta = -1.0$	59,498	91.12	81.27	<b>85.91</b>
SVM	$\theta = -1.1$	97,101	91.49	80.80	85.82
SVM	$\theta = -1.2$	120,856	91.33	80.51	85.58
SVM	$\theta = -1.5$	240,463	92.31	78.01	84.56

# Experiments on Clause Identification



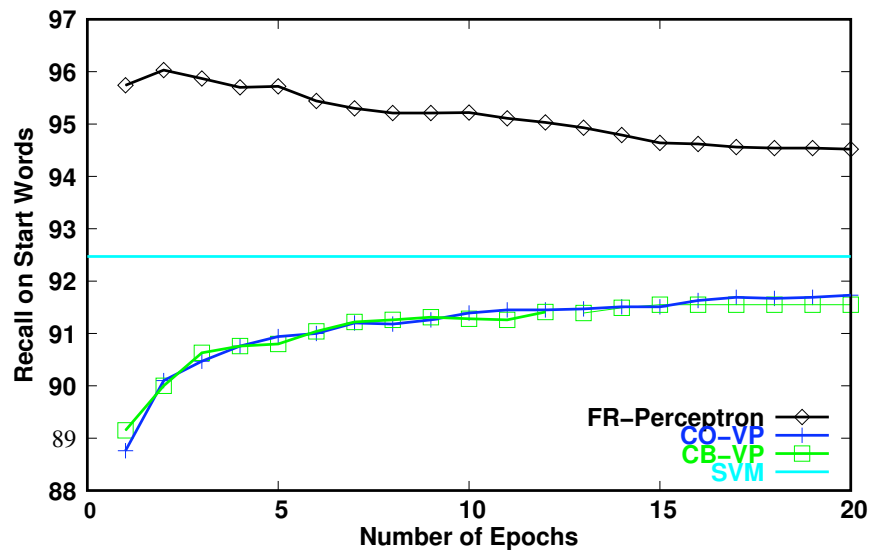
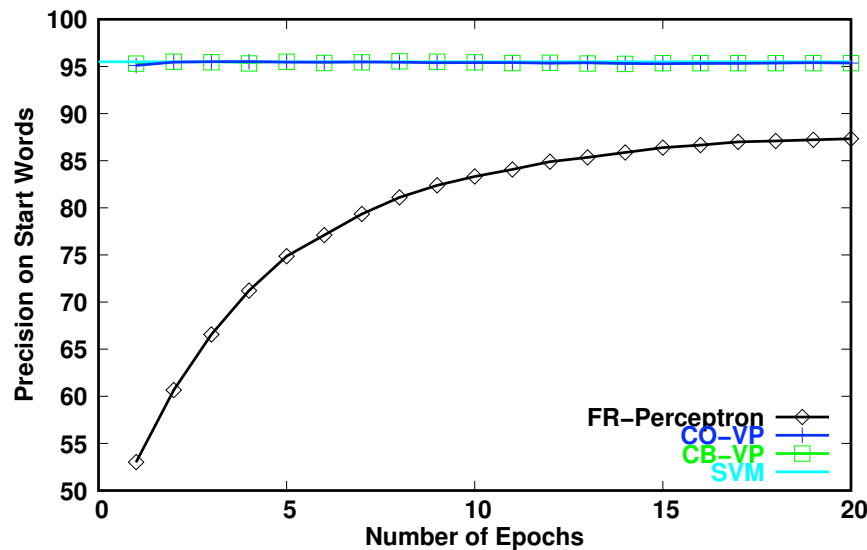
# Results on Clause Identification CoNLL-2001

	T	prec.	recall	$F_{\beta=1}$
goldSE CB-VP	8	82.22	78.09	80.10
goldSE SVM	-	83.19	80.00	81.57
CO-VP	19	89.25	77.62	83.03
learnedSE SVM		91.12	81.27	85.91
FR-Perceptron	20	88.17	<b>82.10</b>	<b>85.03</b>
AdaBoost (1)	-	<b>90.18</b>	78.11	83.71

(1) (Carreras, Màrquez, Punyakanok and Roth, ECML'02)

# Experiments on Clause Identification

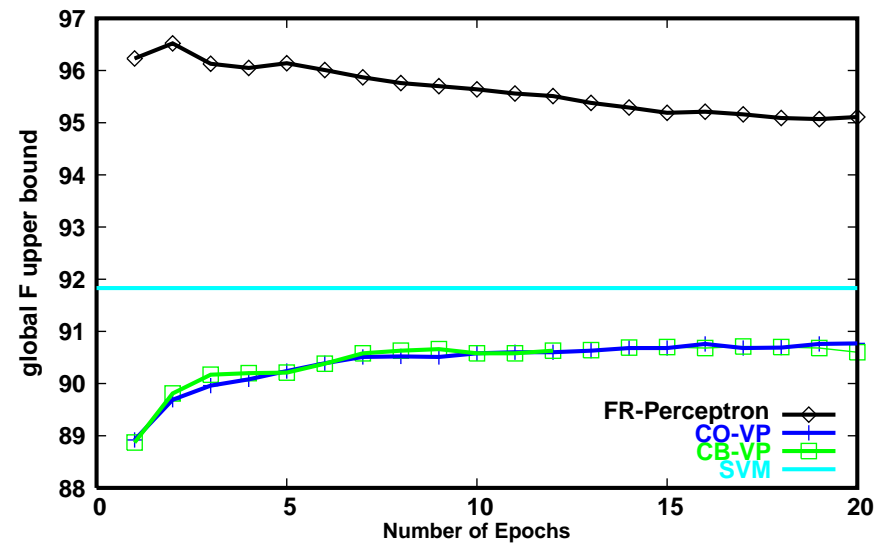
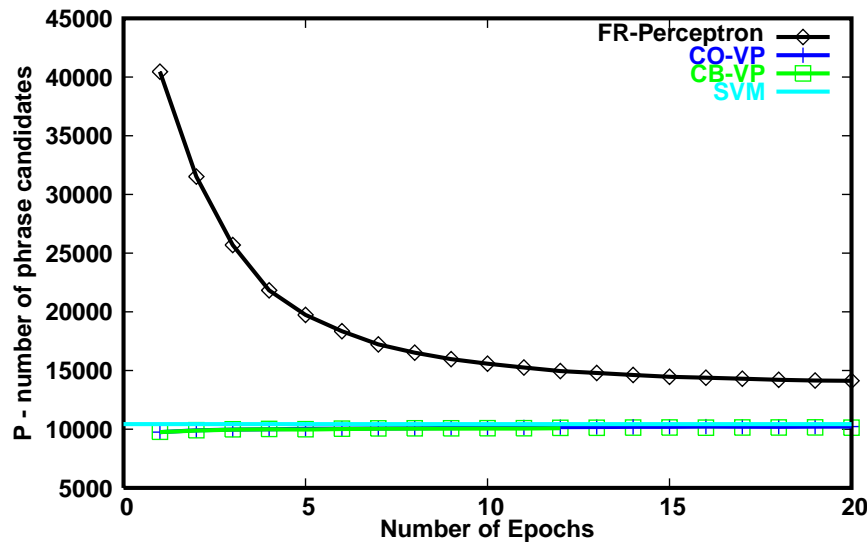
## Precision/Recall on Start words



On End words, a similar behavior is observed.

# Experiments on Clause Identification

## Explored Phrases/Upper Bound $F_1$



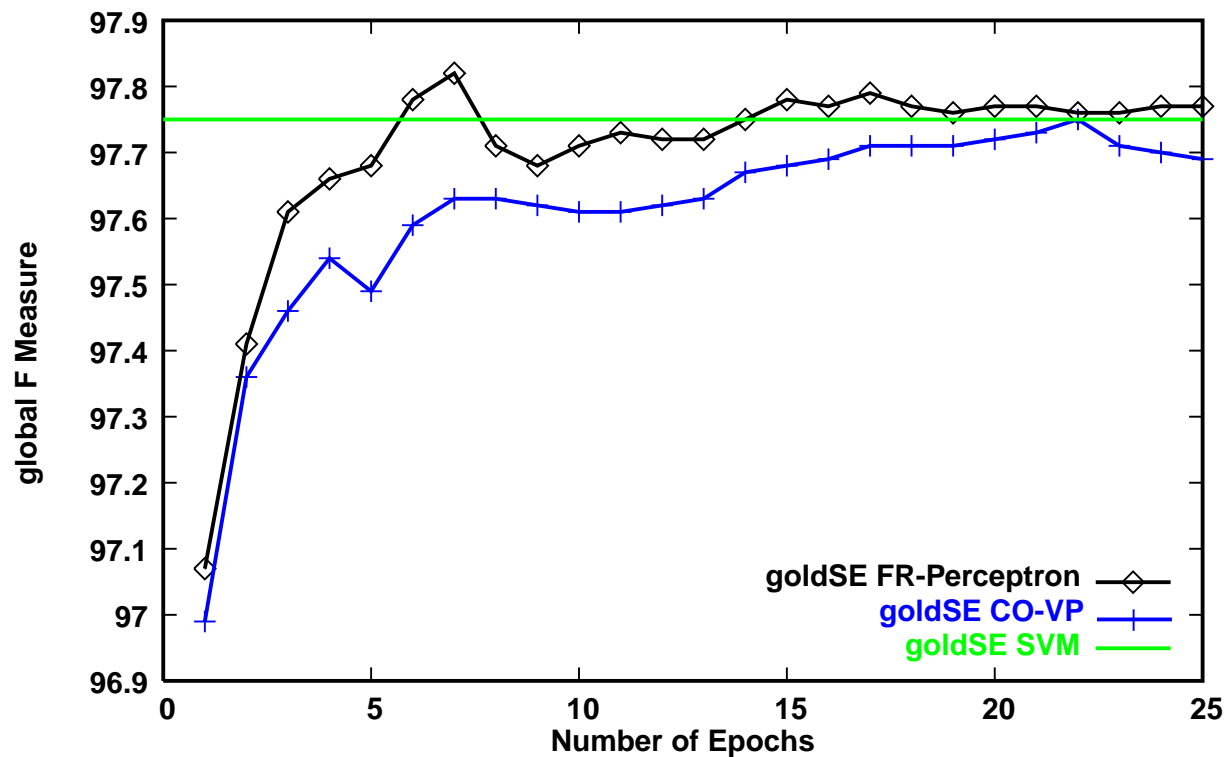
# Experiments on Clause Identification

## Behavior of the Score Function

Three special scenarios in which to experiment with the score function:

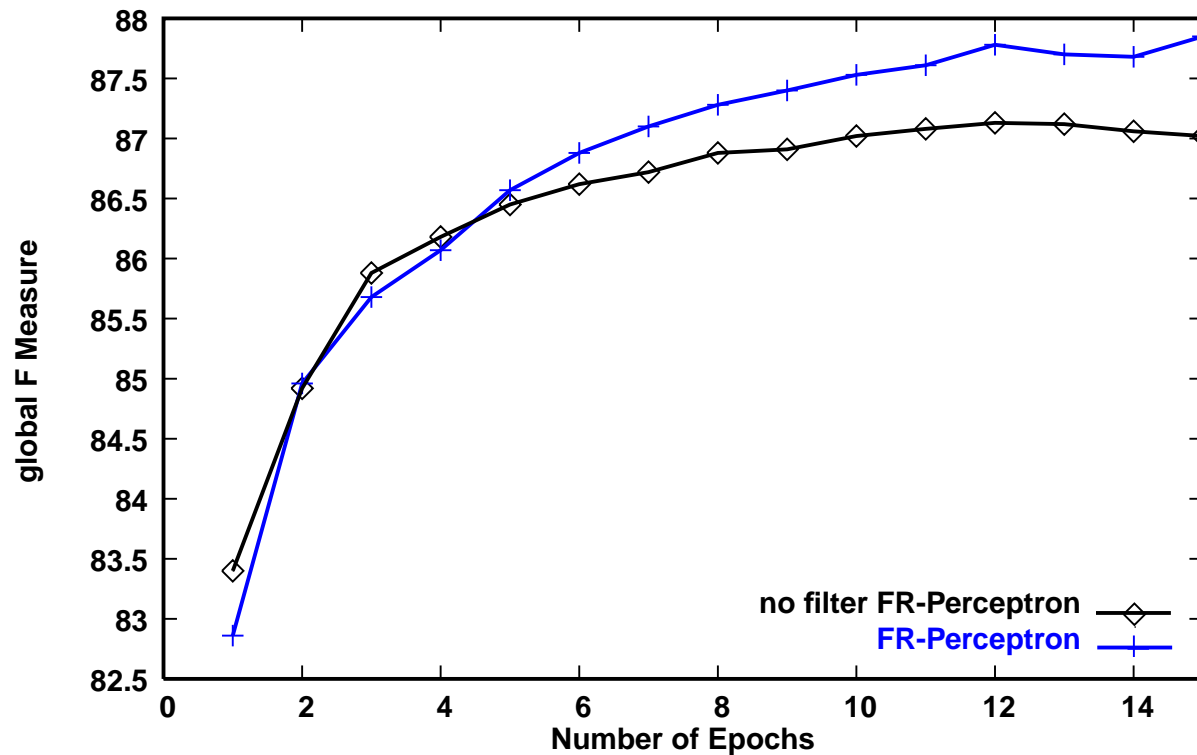
- GoldSE : using the gold filters to train and test.
- No Filtering : all possible phrases are candidates.
- FR Filters: the best obtained filters.

# Experiments on Clause Identification Score Function above Gold Filters

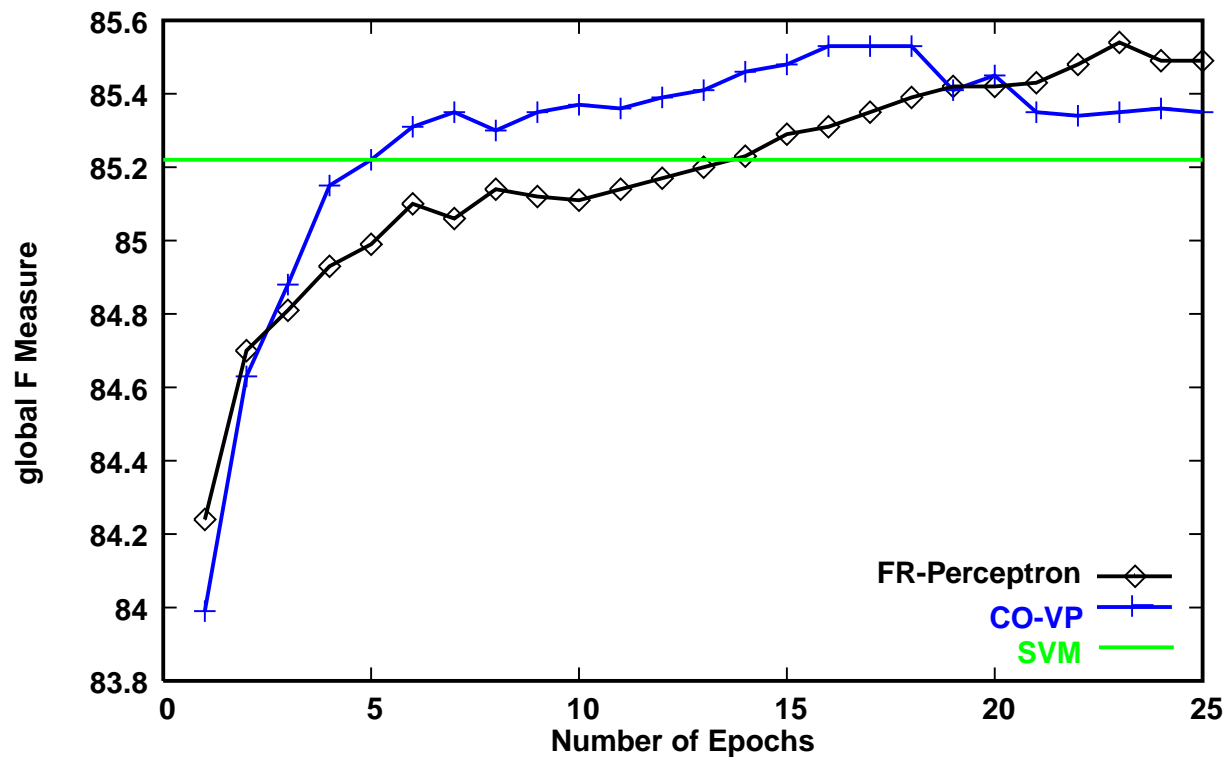


# Experiments on Clause Identification

## Score Function without Filters

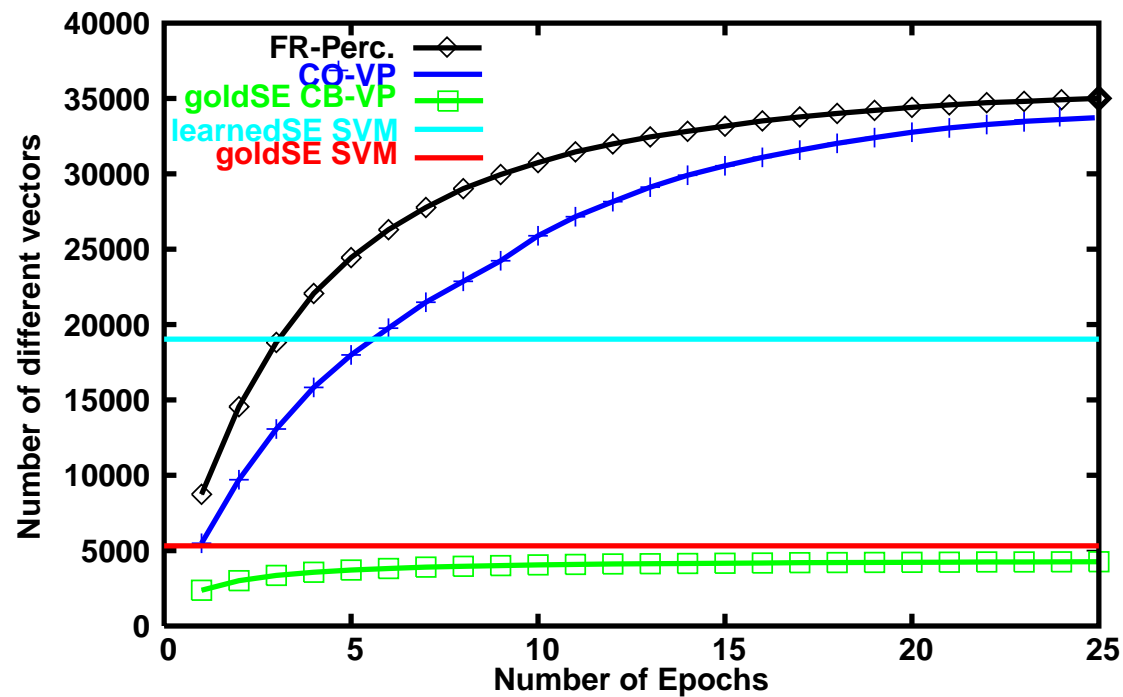


# Experiments on Clause Identification Score Function above FR Learned Filters



# Experiments on Clause Identification

## Practical Problem: number of dual vectors



# Results on Chunking

## 11 types – CoNLL-2000

	technique	prec.	recall	$F_1$
(Kudo & M. 01)	SVM voting	93.89	<b>93.92</b>	<b>93.91</b>
(Kudo & M. 01)	SVM single	–	–	93.85
<b>FRP-Chunker</b>	<b>F&amp;R VP</b>	<b>94.20</b>	93.38	93.79
(Zhang 02)	Winnnow	93.54	93.60	93.57
<b>BI-Chunker</b>	<b>greedy VP</b>	92.83	92.21	92.52

## Results on NP Chunking – CoNLL-2000

Reference	S	Technique	Prec.	Rec.	F <sub>1</sub>
FRP-Chunker	all	F&R VP	94.55	94.37	94.46
(Kudo & M.)	all	SVM voting	94.47	94.32	94.39
(Zhang 02)	all	Winnow (+)	94.39	94.37	94.38
(Sha & Per. 03)	NP	CRF	<i>unav.</i>	<i>unav.</i>	94.38
FRP-Chunker	NP	F&R VP	94.69	93.98	94.33
(Kudo & M.)	all	SVM single	94.54	94.09	94.32
(Sha & Per. 03)	NP	MM-VP	<i>unav.</i>	<i>unav.</i>	94.09
(Zhang 02)	all	Winnow	93.80	93.99	93.89
(Collins 02c)	NP	MM-VP	<i>unav.</i>	<i>unav.</i>	93.53
BI-Chunker	all	greedy-VP	92.83	92.21	92.52

# Semantic Role Labeling

- Recognize predicate arguments and label them according to some scheme (ie. PropBank).
- Same architecture applies:
  - ★ argument = phrase to be recognized
  - ★ We build hierarchies of arguments.
  - ★ Now, the score of a phrase is not only related to its internal structure, but also to a number of predicates.
- 3rd position in CoNLL-2004, not bad!

# Conclusions

- Flexible learning architecture for recovering phrases:
  - ★ The parsing strategy defines the dependencies to be exploited.
  - ★ With Perceptron, the parser functions are easily adapted to work within the parser.
- Future lines:
  - ★ Analysys: understand the global margins.
  - ★ Re-consider NLP pipeline.

**Thanks!**